**Name: Oguguom Lawrence C**

**Matrix: 230908**

**Department: Computer Science**

**ASSIGNMNET CSC235**

**UNIX OPERATING SYSTEM**

Unix is a family of multi-tasking, multiuser computer operating systems that is derived from the original AT&T Unix, whose development started in 1969 at the Bell Labs Research center by Ken Thompson, Dennis Ritchie and others.

It was written in C and assembly language, and it belongs to the Unix Operating System Family and the kernel type is monolithic microkernel, hybrid.

**LINUX- FLAVOR**

A flavor is the variation of the distribution that is being used, which comes down to which window manager (WM) an operator uses and the desktop environment (DE).

It contains many distributions or versions of operating system (OS).

**TEN (10) LINUX DISTRIBUTION OR DATA**

**Ubuntu**

It’s the most popular distro used by far, gathering more than 2,200 hits per day on the distro-watch site alone compared with some 1400 for Fedora. It’s also known as Maverick-Meerkat.

**Fedora**

It’s the free version of Red-Hat whose RHEL (Red Hat Enterprise Linux) has been a commercial product since 2003, it offers a six-month release schedule and it’s security features are excellent.

**Linux-Mint**

This operating system adds to the Ubuntu with it’s own distinct desktop theme and a different set of applications. It comprises graphical tools for enhanced usability such as

1. Mint-Desktop: For configuring the desktop environment
2. Mint-Install: For installing software
3. Mint-menu: For navigating easily

**Opensuse**

This is the foundation for **NOVELL’S SUSE** Linux enterprise desktop.

**PC Linux**

This uses the KDE desktop environment and it’s a lighter weight of weight version of Mandriva and it has no sixty-four (64) bit version.

**Debian**

It has a relatively slow-release cycle with stable ones coming out every one to three years.

**Mandriva**

Mandriva is known as Mandrake and is notable for cutting edge software and 64-bit edition.

**Sabayon/Gentoo**

This is a live **CD** version of Gentoo and it’s known for all allowing users to individual optimize each component.

**Arch Linux Plus Slackware**

This is another distro used primarily for those interested in tweaking and optimizing their system.

**Puppy-Linux Plus DSL**

This ideal for older version hardware and situations where computing resources are minimal

**SOFTWARE FUNCTIONAL REQUIREMNETS**

This are product features or functions that developers must implement to enable the users to accomplish their tasks.

Therefore, it describes system behavior under specific conditions. i.e., it sends an approval request after the user enters a personal information.

**UNIX PREFFERED AT SOME POIINT**

**Power:** Unix provides a range of tools that can be combined and be manipulated to perform such a wide variety of jobs that users of the system can very often carry out tasks without t executing programs in a programming language.

**UNIX PREFERRED TO SCIENTIST AS OPERATING SYSTEM(OS)**

It’s a refreshing alternative to monolithic tools like IDEs and languages like java.

1. The Shell: it has command line interfaces and graphical interfaces living often as servers using text only software reduces the overhead.
2. Everything is a (TEXT) file: They depend on text files contested with other systems of the time that uses opaque binary files to store configuration information.
3. Small tool: It does complex tasks by building pipelines out of small tools. All the shells have a pipeline character, “I” which sends the output of one program into the input of another and this makes stringing together programs easy.

**C AS A PROGRAMMING LANGUAGE**

C is a structural, procedural programming language that has been widely used both for operating system (OS) and applications and that has had a wide following in the academic community.

Many versions of UNIX based operating systems are written in C.

C is also a general purpose programming language that is extremely popular, simple and flexible to use .

The sections of a C program are listed below:

1. Documentation section
2. Preprocessor section
3. Definition section
4. Global declaration
5. Main function
6. User defined functions

### Documentation section

It includes the statement specified at the beginning of a program, such as a program's **name, date, description,** and **title**. It is represented as:

1. //name of a program

Or

1. /\*
2. Overview of the code
3. .
4. \*/

Both methods work as the document section in a program. It provides an overview of the program. Anything written inside will be considered a part of the documentation section and will not interfere with the specified code.

### Preprocessor section

The preprocessor section contains all the header files used in a program. It informs the system to link the header files to the system libraries. It is given by:

1. #inlcude<studio.h>

2. #include<studio.h>

The **#include** statement includes the specific file as a part of a function at the time of the compilation. Thus, the contents of the included file are compiled along with the function being compiled. The **#include<stdio.h>** consists of the contents of the standard input output files, which contains the definition of stdin, stdout, and stderr. Whenever the definitions stdin, stdout, and stderr are used in a function, the statement #include<stdio.h> need to be used.

There are various header files available for different purposes. For example, **# include <math.h>.** It is used for mathematic functions in a program.

Define section

The define section comprises of different constants declared using the define keyword. It is given by:

1. #define a = 2

Global declaration

The global section comprises of all the global declarations in the program. It is given by:

1. **float** num = 2.54;
2. **int** a = 5;
3. **char** ch ='z';

The size of the above global variables is listed as follows:

char = 1 byte

float = 4 bytes

int = 4 bytes

We can also declare user defined functions in the global variable section.

Main function

main() is the first function to be executed by the computer. It is necessary for a code to include the main(). It is like any other function available in the C library. Parenthesis () are used for passing parameters (if any) to a function.

The main function is declared as:

1. main()

We can also use int or main with the main (). The void main() specifies that the program will not return any value. The int main() specifies that the program can return integer type data.

1. **int** main()

Or

1. **void** main()

Main function is further categorized into **local declarations, statements,** and **expressions.**

Local declarations

The variable that is declared inside a given function or block refers to as local declarations.

1. main()
2. {
3. **int** i = 2;
4. i++;
5. }

**Statements**

The statements refers to **if, else, while, do, for**, etc. used in a program within the main function.

**Expressions**

An expression is a type of formula where operands are linked with each other by the use of operators. It is given by:

1. a - b;
2. a +b;

User defined functions

The user defined functions specified the functions specified as per the requirements of the user. For example, color(), sum(), division(), etc.

The program (basic or advance) follows the same sections as listed above.

**Return function** is generally the last section of a code. But, it is not necessary to include. It is used when we want to return a value. The return function returns a value when the return type other than the void is specified with the function.

Return type ends the execution of the function. It further returns control to the specified calling function. It is given by:

1. return;

Or

1. return expression ;

For example,

return 0;

Examples

Let's begin with a simple program in C language.

**Example 1:** To find the sum of two numbers given by the user

It is given by:

1. /\* Sum of two numbers \*/
2. #include<stdio.h>
3. int main()
4. {
5. int a, b, sum;
6. printf("Enter two numbers to be added ");
7. scanf("%d %d", &a, &b);
8. // calculating sum
9. sum = a + b;
10. printf("%d + %d = %d", a, b, sum);
11. return 0;  // return the integer value in the sum
12. }

Output

Structure of a C program

**HOW TO CREATE A C PROGRAMME FILE ON LINUX OS**

### Install Compiler and other Dev tools

To write and execute a C program on Linux, we need a compiler that will compile the code we have written and give us an executable file for the same. Therefore, for that, if you are on Debian or Ubuntu then install build-essential, and on RHEL based distros go for Development Tools.

**For RHEL/Fedora/CentOS**

Run **system update** command, first:

**On Ubuntu or Debian systems**

Sudo apt-get update

Sudo apt-get install build-essential manpages -dev

We need to have sudo or root user access to run the above commands.

### Check GCC version

GCC is the open-source library, which is an acronym for GNU Compiler Collection available for Linux and acts as a compiling system for Program C and other various programming languages, it mainly used to compile C and C++ programs.

### Open a Text editor on Ubuntu

All Linux distros now come with graphical text editors; however, we can use the command terminal to create a text file to write C program codes as well using command line text editors such as **nano, VIM, gedit, and more.**.. Here we are using nano.

To install nano if it is not on your system type:

**Ubuntu**– sudo apt install nano sudo

### Write your first C Program in Linux terminal

Let’s create a **demo C program** in which we will include common C program library **stdio.h** to use various variables including functions for performing input and output. If you are learning C programming, then you already familiar with the **C libraries** that we define in the header of the program to call various functions for performing various tasks.

**For example**, if you are writing a C program that mathematical functions then we need to declare math.h library, for graphics, we include graphic.H and so on…

**Create a file:**.c

nano demo.c

Now, let’s add the following lines to create a simple C program that will give an output “**Say hello to H2s**” when we compile and execute it. To  **Save** the file by pressing **Ctrl+X**, type **Y**, and then hit the **Enter** Key.

### Compile with GCC

1. **Writing a Program to say Hello World**

# include<studio.h>

Into main(){

Printf("Hello World");

return 0;

}

2. **Writing a program to accept the names of ten students**

# include<studio.h>

Strict student{

Char lastname[100]

Char firstname[100]

Char middlename[100]

} stu[10];

Int main(){

Printf("Please Enter Below Information");

Int I;

For(I=0; i<10; ++i){

printf("Enter Student lastname:");

fgets(stu[i]. lastname, sizeof(stu[i]. lastname, stdin);

printf("Enter Student firstname:");

fgets(stu[i]. lastname, sizeof(stu[i].firsttname, stdin);

printf("Enter Student middlename:");

fgets(stu[i]. lastname, sizeof(stu[i].middlename, stdin);

printf("\n");

}

printf("Display Student Information: \n");

for(i=0; i<10; ++i) {

printf(Lastname:");

printf("%s", stu[i].lastname);

printf(Firstname:");

printf("%s", stu[i].firsttname);

printf("Middlename:");

printf("%s", stu[i].middlename);

}

return 0;

}

3. **Program to count from 1 to N**

# include<studio.h>

Int main(){

int num;

printf("Enter Number You Want To Count From");

scanf("%d", &num);

int i;

for(i=0; i<=num; i++){

printf("%d", i);

printf("\n");

}

}

4. **Program That Reverses A Sentence**

#include<studio.h>

void reverse sentence();

int main(){

printf("Please Enter A Sentence:");

reverseSentence();

return 0;

}

void reverseSentence(){

char c;

scanf("%c", &c);

if (c != '\n'){

reverseSentence();

printf("%c

}

}

5. **Program to solve for quadratic equation**

#include<studio.h>

#include<math.h>

int main(void){

double x,y,e,z;

printf("\n Please Enter Value For X:");

scanf("%lf", &x);

printf("\n Please Enter value For y:");

scanf("%lf", &y);

printf("\n Please Enter Value For e:");

scanf("%lf", &e);

Z=(x\*2+y+e);

printf("The Result Of These Values is %lf, z");

printf("\n");

return 0;

}